ONE APPROACH TO VERIFICATION AND VALIDATION OF SIMULATION MODEL

Alexander Mikov[1], Elena Zamyatina[2]

[1]Cuban State University, Faculty of Computer Technologies and Applied Mathematics,
350040 Krasnodar, Stavropolskaya, 149, Russian Federation
[2]Perm State University, Faculty of Mechanics and Mathematics
614990 Perm, Bukireva 15, Russian Federation

e_zamyatina@mail.ru (Elena Zamyatina)

Abstract

The problems of simulation model validation and verification in CAD system Triad.Net are discussed. CAD system Triad.Net is developed for computer system design. It uses simulation as the main method for computer systems investigation. Authors observe the specifications of the simulation model in Triad.Net, consider the program tools for simulation model analysis (information procedures and conditions of simulation) and propose to use them for simulation model validation and verification, debugging and testing. Besides, authors suggest program tools including the intellectual agents and ontology for localization of mistakes determined during verification and validation processes.

Keywords: Simulation Model, Validation, Verification, Debugging, Ontology

Elena Zamyatina is an Associated Professor of Computer Science at Perm State University, the lecturer of the Faculty of Mechanics and Mathematics. Her research interests include simulation as well as parallel and distributed systems. She received a Ph.D. degree in Information and Computer Science in 1993. She has over 30 years of professional experience in programming tools design, particularly in simulation domain. She is the author of over 70 papers in the area of simulation, distributed simulation, parallel and distributed systems

1 Introduction

The primary purpose of this paper is to discuss one of the approaches of scientific information quality enhancing. It is well known that qualifying standards are rather high for scientific information received during some scientific experiment. The validity of scientific information means a high degree of conformity between scientific results and a problem to be solved. It is very important to apply the proper method, to find a corresponding approach, to create a respective mathematic model. And it is actually for simulation and simulation model too. Simulation is one of the most powerful methods for complicated dynamic systems investigations and so it is an actual problem to create valid simulation model in order to receive valid results of simulation experiment during simulation run.

The problems of validity are coupled with problems of debugging and testing. Authors suggest linguistic and program tools for simulation model debugging and testing. These tools and an expert component for mistake localization are considered below. But at first authors discuss the problems of simulation model verification and validity and related works.

2 Verification, Validation and Accreditation

2.1 Problem statement and definitions

The simulation model is a representation or abstraction of something such as entity, or a system. So a model can't be perfect because it is an abstraction. But we intend to build a credible simulation model and to receive the credible results of simulation. Both the modelers and users of simulation model are interested in reliable simulation results because it is very important to accept the right decision.

Verification and validation (V&V) are considered usually as a single process of M&S (Modeling and Simulation) but it is not accurate reasoning because each of them purposes on the different aim.

The aim of verification is to be sure that simulation model implementation is proper, to determine whether it corresponds to conceptual description and specification. Simulation model verification "is substantiating that the model is transformed from one form into another, as intended, with sufficient accuracy. Model verification deals with building the model right"[1].

The aim of validation is to determine the degree to which a model is an accurate representation of a real system from the perspective of intended use of the model. Model validation "is substantiating that the model, within its domain of applicability, behaves with satisfactory accuracy consistent with the M&S objectives. Model validation deals with building the right model"[1].

In addition we must say about model debugging and testing. Model debugging supposes the detection of mistakes, its localization and elimination. Simulation model testing "is ascertaining whether inaccuracies or errors exist in the model"[1].

We can mention one more definition coupled with verification and validation. It is accreditation – a statement of M&S sponsors that simulation results are intended for use. Accreditation is "the official certification that a model or simulation is acceptable for use for a specific purpose"[2].

2.2 Related works

V&V is well known problem. It is discussed in many papers. So one may become acquainted with these problems thanks to numerous publications of O.Balci[1,3] A.Law[4], R.Sargent[5,6] and the other authors.

These papers present different paradigms of verification and validation, define different stages, give recommendations and consider methods but most of them don't show how to do it.

2.3 V&V in Triad

Let us consider the simulation model validation more precisely. It is well known that V&V must be fulfilled on different levels (input data, simulation model elements, simulation model subsystems and its interconnections).

The testing of simulation model adequacy and accuracy includes it's structure testing (one must determine whether the structure of the simulation model, a list of objects and their interconnections correspond to investigator's intentions, let us name this type of validity as the structural one), primitive functions testing, behavior testing (one must examine whether the simulation model functionality corresponds to investigator's concepts. Let us name this type of validity as an operational one). Moreover, simulation model validation has to be executed at each stage of simulation model design. It is necessary to return to the previous stage if the process of simulation model validation shows some errors.

Structural simulation model validity may be fulfilled by functions and procedures of Triad-model structure layer. These functions and procedures may test the topological specification of a simulation model in particular. But operational validity may be carried out by the information procedures. Information procedures allow to determine whether any value of simulation model variables at some concrete moment of simulation time is equal to the specific one in right (valid) model and so on. (We shall discuss the information procedures possibilities more precisely below). If it is not so (the values are not equal) then the cognitive agents will define the type of mistake

using specific ontology and localize it following some rules which use knowledge about simulation model structure and behavior. Let us consider the specification of simulation model in Triad. We must remind that CAD Triad system is intended for computer systems design and analyses and therefore it has some specifications.

3 Simulation Model in Triad

3.1 Triad-model representation

Program model in Triad.Net is represented by several objects functioning according to some scenario and interacting with one another by sending messages. Program model [7] is μ={*STR*, *ROUT*, *MES*} and it consists of three layers, where *STR* is a layer of structures, *ROUT* – a layer of routines and *MES* – a layer of messages appropriately. The layer of structure is dedicated to describe objects and their interconnections, but the layer of routines presents their behavior. Each object can send a message to another object. So, each object has the input and output poles (P_{in} – input poles are used to send the messages, P_{out} – output poles serve to receive the messages). One level of the structure is presented by graph $P = \{U, V, W\}$. P-graph is named as graph with poles. A set of nodes V presents a set of programming objects, W – a set of connections between them, U – a set of external poles. The internal poles are used for information exchange within the same structure level; in contrast, the set of external poles serves to send messages to the objects situated on higher or underlying levels of description. Special statement ***out*** <message> ***through*** <name of pole> is used to send the messages.

One can describe the structure of a system to be simulated using such a linguistic construction:

structure <*name of structure*> ***def*** (<*a list of generic parameters*>) (<*a list of input and output parameters*>) <*a list of variables description*> <*statements*>) ***endstr***

Special algorithms (named "routine") define the behavior of an object. It is associated with particular node of graph P = {U, V, W}. Each routine is specified by a set of events (E-set), the linearly ordered set of time moments (T-set), and a set of states {Q-set}. State is specified by the local variable values. Local variables are defined in routine. The state is changed if an event occurs only. One event schedules another event. Routine (as an object) has input and output poles (Pr_{in} and Pr_{out}). An input pole serves to receive messages, output – to send them. One can pick out input event e_{in}. All the input poles are processed by an input event, an output poles – by the other (usual) event.

routine<имя>(<*a list of generic parameters*>)(<*a list of input and output formal parameters*>) ***initial*** <*a sequence of a statements*> ***endi***
event <*a sequence of a statements*> ***ende***
event <*a name of an event*> <*a sequence of statements*> ***ende*** …
event<*a name of an event*><*a sequence of a statements*> ***ende endrout***

Let us present the text of Triad model description:

```
Type Router,Host; integer I;
        M:=dStar(Rout[5]<Pol[4]>);
        M:=M+node Hst[8]<Pol>;
       M.Rout[0]=>Router;
  for i:=1 by 1 to 4 do
              M.Rout[i]=>Router;
      M:=M+edge(Rout[i].Pol[1]— Hst[2*i-2]);
      M:=M+edge(Rout[i].Pol[2] —Hst[2*i-1]);
  endf;
  for i:=0 by 1 to 7 do
              M.Hst[i]=>Host;
  endf ;
```

Fig. 1 The fragment of Triad-simulation model

Let us present a simulation model in Triad (one can pick it out at fig.1.). It is a fragment of computer network (fig.2.).

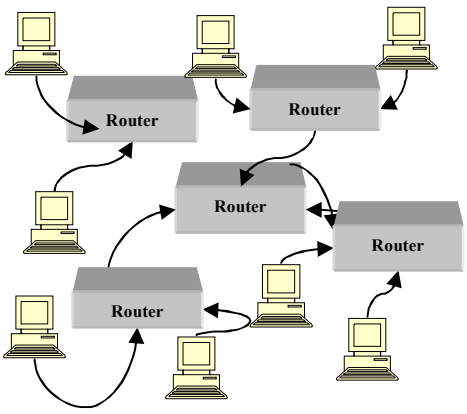

Fig. 2. Computer system for modeling and simulation

Computer system consists of some workstations and routers and provides message sending and receiving. The structure of simulation model is described by graph constant ***dstar*** with 5 nodes associated with routers. Eight nodes represent workstations Hst[i]. These nodes are added in cycle. Each router is intended to fulfill the same algorithm. So designer defines the same routine Router (appropriate program module is saved in data base) for each router (node Rout[i] of structure) and the same routine Host to each workstation (node Hst[i]).

A program model in Triad.Net isn't static. Triad language includes the special type of variables – type "model". There are several operations with the variable type "model". The operations are defined for the model in general and for each layer. For example, one may add or delete a node, add or delete an edge (arc), poles, union or intersection of graphs. Routine layer permits to add or delete any event, layer of messages – to add or delete types or selectors. Besides, one or another routine can be assigned to the node in structure layer (using some rules). As a result the behavior of the object associated with this node would be changed.

3.2 Algorithm of investigation

The objects of simulation model are managed by the special algorithm during the simulation run. Let us name it as "simulation algorithm" (CAD system Triad has distributed version and corresponding algorithm for distributed objects of simulation model too) [8]. CAD system Triad includes analyses subsystem implementing the algorithm of investigation - special algorithm for data (the results of simulation run) collection and processing.

The analysis subsystem includes special objects of two types**:** *information procedures* and *conditions of simulation*. Information procedures are "connected" to nodes or, more precisely, to routines, which describe the behavior of particular nodes during simulation experiment. Information procedures inspect the execution process and play a role of monitors of test desk.

Conditions of simulation are special linguistic constructions defining the algorithm of investigation because the corresponding linguistic construction includes a list of information procedures which are necessary for investigator.

The algorithm of investigation is detached from the simulation model. Hence it is possible to change the algorithm of investigation if investigator would be interested in the other specifications of simulation model. For this one need to change the conditions of simulation. But the simulation model remains invariant. We may remind that it is not possible in some simulation systems.

One can describe the information procedure as so:

information procedure<*name*>(<*a list of generic parameters*>)(<*input and output formal parameters*>)
initial <*a sequence of statements*> ***endi***
<*a sequence of statements*>***processing*** <*a sequence of statements*>…***endinf***

It is possible to examine the value of local variables, the event occurrence and the value of messages which were sent or received. A part of linguistic construction 'processing' defines the final processing of data being collected during simulation run (mean, variance and so on).

Let us present the linguistic construction ***conditions of simulation***:

Conditions of simulation<*name*>(<a list of generic parameters>)(<input and output formal parameters>) **initial** <a sequence of statements> ***endi*** <a list of information procedures> <a sequence of statements> ***processing*** <a sequence of statements>…***endcond***

The linguistic construction ***conditions of simulation*** describes the algorithm of investigation which defines not only the list of information procedures but the final processing of some information procedure and checks if conditions of simulation correspond to the end of simulation.

simulate <a list of an elements of models, being inspected> ***on conditions of*** simulation <name> (a list of actual generic parameters>)[<a list of input and output actual parameters>]

Simulation run is initialized after simulation statement processing.

One can pay an attention to the fact that the several models may be simulated under the same *conditions of simulation* simultaneously.

The subsystem of visualization represents the results of simulation.

4 Information procedures for simulation model validation

4.1 Error detection

Information procedures are convenient not only for simulation model analyses but for simulation model debugging and validation too. Primarily we shall consider the types of errors which may be recognized by the information procedures.

So is: incorrect temporary delays, wrong messages transfer and semantic incorrectness of data exchange, invalid management of simulation model functioning, semantic incorrectness of changing of states of simulation model, forbidden simulation model states.

It is advisable to determine the correctness of the following values for simulation model being represented above: time slice between sending of message from one workstation and receiving of message by another one (this time slice must be less than some limiting value). Moreover it is important to be sure that the value of received message is valid and it is received by the correct input of workstation.

CAD system Triad has a set of standard information procedures for temporary delays recognition. These procedures serve as a basis of knowledge based debugger. Information procedure *more_interval (t) [e1, e2]*, for example, is able to fix time slice between two events e1 (maybe it is the event of message sending) and e2 (the event of message termination).

Invalid temporary delays may be recognized by information procedures *all_events(e)* (this procedure defines the particular beginning of event and its termination), *all_changes(var)* (procedure defines each instant of the time when indicated variable is changed) and so on.

Investigator may use information procedure *schedule_event (e)* (the names of all registered events which preceded the indicated one); *inspect_change (e)* (the values of all variables which were changed before the event *e* occurrence, event *e* is an actual argument of procedure).

So we named some standard information procedures. But an investigator may use linguistic constructions (information procedure and conditions of debugging) for specific occasions of debugging and so he will share the debugger functionality.

Let us consider the example of information procedure for the detection of the sequence of events arrival:

information procedure EVENT_SEQUENCE (***in ref event*** E1,E2,E3;***out Boolean*** ARRIVED)
initial interlock (E2,E3); ARRIVED := false;
case of E1:available(E2);
E2:available(E3):
E3:ARRIVED:=true;
endc
endinf

So investigator may detect the arrival of the sequence of events E1→E2→E3. The statement ***interlock*** provides input parameter blocking (event E1 in this case). It means that information procedure doesn't watch parameters being marked in interlock statement. The statement ***available*** allows beginning the marked parameter monitoring again.

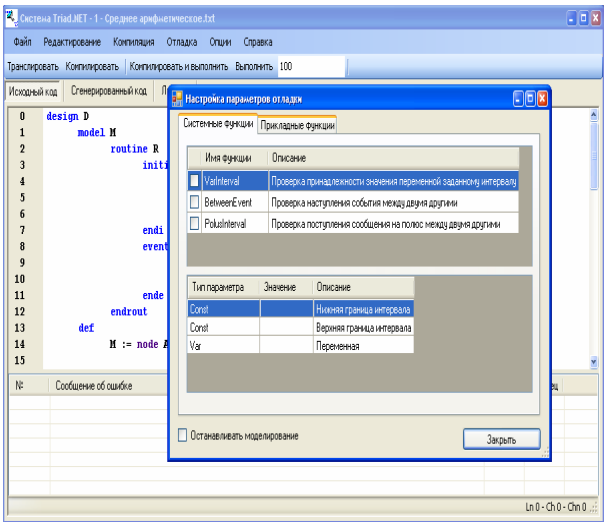

Fig 3. Simulation model editor and information procedures for algorithm of investigation

Another example concerns the problem of forbidden states of simulation model detection. Only such linguistic and programming tool as information procedure may detect this error because only information procedure may determine the value of local variables of different routines at the same moment of simulation time during simulation experiment and may compare these values.

Simulation model editor and information procedures for data collection and model monitoring are presented at fig.3.

4.2 Knowledge based debugger

Thus we have discussed the problem of error debugging and testing and consider the linguistic and program tools in Triad intending to solve these problems. We briefly presented information procedures, its linguistic construction and examples of applying these tools for simulation model monitoring during the simulation run. Special linguistic construction *conditions of debugging* contains a list of the information procedures which are intended for simulation model monitoring and error detection. The construction *conditions of debugging* are a part of simulate statement. Debugger starts its work when simulation model begins to fulfill the *simulate* statement.

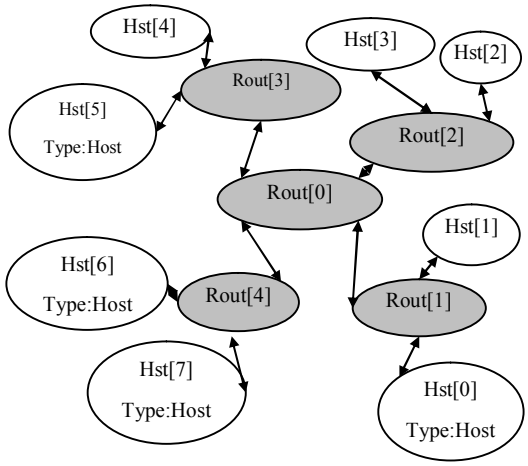

Fig. 4. Graph representation of the simulation model structure

Let us consider such a situation: we want to be sure in data transfer correctness from the workstation Hst[0].Pol[1] (Pol[1]-it is an output polus) to workstation Hst[4].Pol[1] (fig.4). The example of simulate statement one can see below:

simulate M ***on*** Checker (M.Hst[0].Pol[1], M.Hst[4].Pol[1])

Corresponding linguistic construction **conditions of debugging** is represented below:

conditions of debugging Checker (*input real* a,b; ***output boolean*** answer) (…Check_Value(1.0,1.0) (***input*** ia,ib; ***output*** ianswer);…).

The simulation run has to be terminated because of error detection. It is necessary to localize error and neutralize it. Error localization and neutralization is knowledge based.

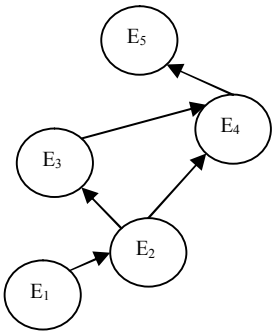

Fig. 5. Graph of scheduled events

Let us return to the example (fig. 4). We shall suppose that the routine associated with node Hst[0] sends a signal to routine associated with HST[4]. Let debugger detects invalid signal received at input of this routine. So this invalid signal may be the result of invalid processing in routines Rout[1], Rout[0], Rout[3]. Moreover it is advisable to check the sequence of events and temporary delays validity.

So the mistake localization may develop according to different scenarios (may conform to different rules). These rules are contained in knowledge base. More precisely cognitive agents for each type of the mistake with appropriate rules are used for mistake localization. Debugger needs additional information: simulation model structure which is represented by P = {U, V, W}. This information is necessary to define the path message transfer. Besides, the information mentioned above is helpful to use the graph of scheduled events in order to determine the sequence of events participating in message processing (it is presented on fig.4). Debugger may use such additional information as graph of calculations. This graph provides the determination of sequence of local variables processing.

Expert system for mistake localization includes a set of procedures for debugging, knowledge base with rules 'if….then …' using all additional information mentioned above. Besides, it includes inference engine, explanation module, editor for rules and meta rules. A set of procedures for localization of mistakes may be extended because an investigator may create new information procedures using appropriate linguistic tools.

It is necessary to tell about ontology which may be used for mistake detection. Debugger will form request to ontology if the mistake of the specific type is detected. Appropriate cognitive agent with specific rules needed for an algorithm of processing mistake of specific type will be started. If there are some paths of an algorithm then some cognitive agents will be started.

5 Conclusion

So the authors presented the linguistic and program tools of CAD Triad.Net system needed not only for design of model but for its monitoring, data collections, analyses, validation and verification. The authors consider the linguistic and intellectual program tools for simulation model verification and validation more precisely. The appropriate linguistic tools - information procedures - are under discussion.

The architecture of the debugger is presented. The debugger is used not only to detect mistakes in the simulation model but to localize them. The debugger detects mistakes using information procedures and determines the appropriate rules of localization thanks to ontology. Proper cognitive agents which act in accordance with these specific rules start to localize mistakes. So the authors suggest solving the problem of validation and verification using multiagent approach and ontology.

So presented approach permits to automate simulation model verification and validation. The process of mistake detection becomes more effective and more flexible.

The problem of simulation model testing and more detailed developmental work of multiagent subsystem are under consideration of authors nowadays.

6 References

[1] Balci O. Verification, Validation, And Accreditation. Proceedings of the 1998 Winter Simulation Conference. D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, eds.. Grand Hyatt Washington, Washington DC, pp.41-48.

[2] Department of Defense. 1996. Department of Defense Verification, Validation and Accreditation (VV&A) Recommended Practices Guide, Defense Modeling and Simulation Office, Alexandria, VA, Nov. (Co-authored by: O. Balci, P. A. Glasow, P. Muessig, E. H. Page, J. Sikora, S. Solick, and S. Youngblood) http://triton dmso.mil/docslib/mspolicy/vva/rpg/

[3] Balci O, Nance R.E., Arthur J.D. Expanding Our Horizons In Verification, Validation And Accreditation Research And Practice Proceedings of the 2002 Winter Simulation Conference E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds., WSC 2002, San Diego, California, pp.653-663.

[4] Law, A. M. and M. G. McComas. 2001. How to build valid and credible simulation models. In Proc. 2001 Winter Simulation Conf., ed. B.A.

Peters, J. S. Smith, D. J Medeiros, and M. W. Rohrer, Piscataway, New Jersey: IEEE. pp. 22-29.

[5] Sargent, R. G. 2005. Verification and validation of simulation models. In Proc. 2005 Winter Simulation Conf., ed. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 130-143. Piscataway, New Jersey: IEEE.

[6] Sargent, R. G. Verification And Validation of Simulation Models. In Proc. 2007 Winter Simulation Conf., S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds., J.W. Marriott Hotel, Washington, D.C., pp.124-137

[7] Mikov A.I. Formal Method for Design of Dynamic Objects and Its Implementation in CAD Systems // Gero J.S. and F.Sudweeks F.(eds), Advances in Formal Design Methods for CAD, Preprints of the IFIP WG 5.2 Workshop on Formal Design Methods for Computer-Aided Design, Mexico, Mexico, 1995. pp.105-127.

[8] Миков А.И., Замятина Е.Б. Проблемы реализации системы распределенного моделирования с удаленным доступом. «Методы и средства обработки информации», Труды третьей Всероссийской научной конференции, 6-8 октября 2009 г.Москва, МАКС ПРЕСС, 2009. стр. 38-44.